The Grid Observatory

Cecile Germain-Renaud, Alain Cady, Philippe Gauron, Michel Jouvin, Charles Loomis, Janusz Martyniak, Julien Nauroy, Guillaume Philippon, Michèle Sebag

The Grid Observatory

Cécile Germain-Renaud*, Alain Cady*, Philippe Gauron*, Michel Jouvin†,
Charles Loomis†, Janusz Martyniak‡, Julien Nauroy*, Guillaume Philippon†, Michèle Sebag*
* Laboratoire de Recherche en Informatique, CNRS − INRIA − Université Paris-Sud 11
†Laboratoire de l'Accélérateur Linéaire, CNRS −Université Paris-Sud 11
‡London Imperial College

***Abstract*—The goal of the Grid Observatory project (GO) is to contribute to an experimental theory of large grid systems by integrating the collection of data on the behaviour of the flagship European Grid Infrastructure (EGI) and its users, the development of models, and an ontology for the domain knowledge. The GO gives access to a database of grid usage traces available to the wider computer science community without the need of grid credentials. The paper presents the architecture of the digital curation process enacted by the GO and examples of their exploitation.**

I. Introduction

The e-Infrastructures actions worldwide foster the emergence of new research environments in which virtual communities of scientists and engineers are empowered to share and exploit the collective power of the ecosystem of scientific and engineering facilities. Ironically, the only community for which this emergence is still to come is Computer Science and Engineering (CSE), and more specifically its sub-segment concerned with Distributed Computing Infrastructures (DCI). While all other major scientific areas of e-science (hard, soft, and humanities) benefit from comprehensive and well curated datasets, these are not mainstream in CSE-DCI.

The goal of the Grid Observatory project (GO) is to contribute to an experimental theory of large grid systems by integrating the collection of data on the behavior of the European Grid Infrastructure (EGI) and its users, the development of models, and an ontology for the domain knowledge. The GO targets the major components of *digital curation* as summarized in [25]: it develops long-term repositories of digital assets for current and future references, provides search and retrieval facilities to scientific communities through its portal, and adds value to data by generating new sources of information and knowledge. The GO has been in operation since October 2008, continuously recording traces and running a production service for the overall CSE community through its portal www.grid-observatory.org.

Section II presents EGI, the flagship grid infrastructure project funded by the European Commission.

Extensive monitoring tools have been developed and put in production during the EGEE/EGI project, providing a unique observational facility. The first role of the Grid Observatory is to preserve these monitoring data, which were previously discarded after operational usage, and to make them available to the scientific community. Adding knowledge starts with the conversion of the traces to standardized representations. Section III details the available traces, the distributed acquisition architecture, and exemplifies two data representation strategies.

The second role of the Grid Observatory is to contribute to advancing issues that are critical for the future of the EGI, for instance scalable resource allocation or dependability. Previous work, for instance [16], [26], [4] exploited the GO data in each of these area, with for some of them the goal of switching from procedural implementations to autonomic self-optimization. Section IV presents two new self-contained examples. The first one illustrates the growing concern for extreme behavior in grid context; the second one exemplifies an often-underestimated difficulty: for many questions of real interest in grids and clouds, no reference and quantifiable interpretation (analogue of labels in classification) is known.

An essential achievement of the GO is to cover the complete scope of the grid middleware and users activity, which goes beyond particular aspects, such as the basic job lifecycle, or the failure events. Section V discusses the relations of the the GO with other relevant initiatives.

II. The EGI grid

A. Overview

The EGI grid infrastructure currently federates computing resources from 317 sites in 52 countries with 243000 CPU cores and 70 PetaBytes of disk storage available to its users. Around 10000 users from variety of scientific domains take advantage of the infrastructure, running on average of 15 million jobs per month.

EGI combines these globally distributed resources into a single production infrastructure. Each participating site configures, runs, and maintains a batch system containing its computational resources and makes those resources available to the grid via a gatekeeper. Similarly, a site's storage resources are accessible to the grid infrastructure via a grid service running a Storage Resource Manager (SRM) interface. Each site defines its own scheduling, quotas, and access policies; the overall policies for EGI as a whole emerge implicitly from the local policies.

One of the three core middlewares for EGI is gLite [9], which functions much like a large distributed batch system. However, this batch-like architecture is not the only possible usage scenario. Many Virtual Organizations (VOs) deploy additional services over gLite services to cope with gLite limitations, hide details from users, provide better prioritization,

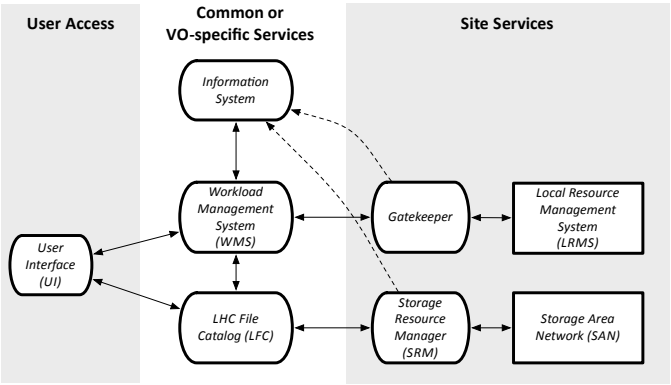

Fig. 1. Simplified gLite Architecture

or manage faults. Schedulers like DIANE [22] use placeholder jobs ("pilot jobs" [18]) in gLite to achieve these goals. In this case, all of the usage is monitored by gLite and information about that activity is collected normally. However other systems like DIRAC [21] are capable of accessing resources outside of the grid; such resource utilization does not appear in the gLite monitoring and is out of scope for the work described here.

Figure 1 shows a simplified architecture of the gLite middleware. Users access the grid infrastructure from "User Interfaces" (UI), machines (ideally someone's workstation) with the grid client software installed. The client software allows access to computing and storage resources on participating sites through grid services, which are described in the following sections. The Information System (IS) ties the whole system together, allowing users to discover services and to view the approximate state of the entire grid infrastructure.

B. Data Management Services

1) Site Storage Services: Site managers typically deploy a collection of disks or a Storage Area Network (SAN) to provide storage for grid users. A grid service, the Storage Resource Manager (SRM), provides a uniform interface for grid users to localize particular files stored at the site. There are multiple implementations of SRM on the grid: DPM, dCache, and SToRM. Multiple services provide direct access to identified files through various protocols: GridFTP, http, rfio, xrootd, etc. The GridFTP protocol is universally supported and heavily used. Information about transfers of files through the GridFTP protocol is included in the Grid Observatory database.

2) LHC File Catalog (LFC): The LFC provides a complete catalog of all of the files stored on the grid for a particular Virtual Organization. Data management services interact with the LFC to register new files on the grid and to locate the "closest" (in network space) replica. The WMS also interacts with this catalog to co-locate calculations with needed input data.

C. Computing Resource Services

1) Site Computing Resources: Resource centers have traditionally used batch systems (or Local Resource Management System, LRMS) to make computing resources available to a large number of users and to manage access (queuing, fair share, quotas, etc.) for them. The gLite middleware builds on this, adding a "gatekeeper" service to act as the bridge between grid users and the local batch system. This provides a uniform interface for grid users and the flexibility for system administrators to use batch systems with which they are familiar. The abstraction for the computing resources is the Computing Element (CE) [12].

2) Workload Management System: At the core of the gLite middleware is the Workload Management System (WMS), a metascheduler, that selects (matchmaking [19]) appropriate computing services based on high-level job requirements provided by the user in a Job Description Language (JDL) and the (approximate) current state of the available resources as provided by the Information System (IS). This service consists of a number of separate daemons to manage the job workflow. The two most important daemons in this context are the logmonitor, which provides detailed information on the scheduling of jobs, and the Logging and Bookkeeping (L&B) service [11], which records all of the significant events in the job's lifecycle gathered from other WMS daemons and site services. The information about the overall job lifecycle in the GO comes from the L&B.

To allow the system to handle the workloads on the grid infrastructure, multiple instances of the WMS are deployed. Each instance caches service state information taken from the grid information system and acts *independently* from all other WMS instances. Because there is no direct coordination between the WMS instances, sub-optimal scheduling and interference can occur.

III. The Grid Observatory archive and portal

The GO portal gives access to a database of grid usage traces available to the wider computer science community. These data are stored on the grid, and made accessible through a web portal without the need of grid credentials. This section goes bottom-up. It first presents the organization of acquisition and publication. Then, the raw traces are described, the goal being to show how the various aspects of the middleware and users activities are covered. Finally, the models proposed for data representation are discussed.

A. Architecture

There are two aspects to the Grid Observatory: data collection and publication. The architecture of the system mirrors those components. Figure 2 provides a graphic overview of the system.

For the data collection, a set of agents collects raw data from the targeted grid systems; the access and transfer methods vary for each system. For the LRMS, WMS, and gatekeeper, the service logs are obtained via secure (because they are not yet anonymized) ftp from the running systems. Data from the Logging and Bookkeeping service are obtained directly from the backing relational database. As the Information System (IS) is based on LDAP, standard LDAP commands are used

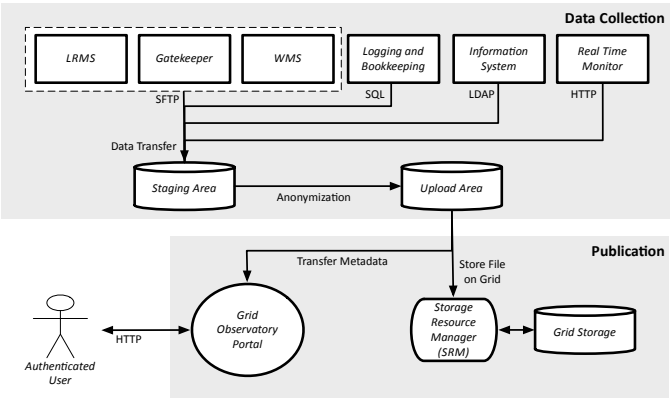

Fig. 2. Grid Observatory Architecture

to collect and save the grid's published state. Finally the Real Time Monitor (RTM) project, independent of the GO, collects a subset of information from a large number of instrumented WMS instances. This information is made available to the GO through an HTTP connection. The agents periodically collect the information from the services to allow the grid's evolution over time to be studied.

All of the data are written to a staging area, where a separate set of scripts anonymize the content. Once this process is complete, the data are pushed back onto the grid for storage using the standard grid client tools and services. These data are stored within the Grid Observatory's VO and members of that VO have direct access to the anonymized files via the grid.

A separate web portal, associated with a database of meta-data about the published files, is the entry point for the general users to search for and to access these data. It would indeed be too restrictive to require all CSE users to obtain grid credentials, register with the VO, find an appropriate user interface in order to access the data. Instead, the users must first go through a lightweight registration mechanism to protect against malicious access. Following their registration, the portal permits users to access the Grid Observatory data with a username/password pair. The portal acts as a proxy for data access requests, using the portal's grid credentials to access the data on the grid and subsequently transferring the data to the authenticated user. The web interface allows for selecting the files from the metadata, currently dates and kind of traces.

B. The traces

Table I summarizes the information about the available traces. Depending of the middleware component logged, the *Range* may be *Comprehensive* (reporting on the whole EGI grid), *Partial* (a cross-cut of the whole activity), or *Local* (reporting on a site). Orthogonally, the *Scope* of the traces may be limited to gLite monitored information, or report on all EGI. Currently, the GO provides the traces in their native format; a few of them follow actual or de-facto standards, but most formats are gLite-specific (*Spec.* in the table). The decision of

TABLE I
THE GRID OBSERVATORY TRACES.

Component	Range	Scope	Format	Size
RTM	Comprehensive	gLite	Spec.	200MB
IS	Comprehensive	EGI	LDIF	300MB
L&B	Partial	gLite	SQL	2GB
Accounting	Local	gLite	PBS	6MB
CondorG	Partial	gLite	Spec.	15 KB
JobController	Partial	gLite	Spec.	40MB
LogMonitor	Partial	gLite	Spec.	70 MB
WorkloadManager	Partial	gLite	Spec.	70MB
GridFTP	Partial	EGI	Spec.	11MB

providing the raw traces is discussed in section III-C. The GO documentation [3] provides a complete syntactic and semantic description of the traces, which is only sketched here.

Each trace has its own recording frequency; however, if this frequency is higher than weekly, the individual files are collected into a week-comprehensive directory, which is provided as a compressed archive. The typical volume of each weekly compressed archive is indicated in the last column of the table; of course, much fluctuation is possible depending on the grid activity.

1) Real Time Monitor (RTM) traces: they are provided by the RTM project [7]. The RTM summarizes information that is available from all the Logging and Bookkeeping databases. The RTM is able to show various information such as running and scheduled jobs, job transfers and detailed information on Resource Brokers and Computing Elements for each site. This information, together with the IS archives published by the GO, should allow exploring many of the questions related to grid usages, as long as only jobs are considered.

The trace registers 37 attributes for each job, which can be distinguished as categorical (table II), timestamps (table III), and metrics (no described here as they are derived from the timestamps). Each job record is a tab-separated attribute list inside a line. The trace is provided as a weekly archive. Each file in the archive corresponds to the activity of one CE and one day.

2) Information System (IS): it provides detailed information to the grid services about the static and dynamic status of the grid infrastructure and services. Amongst tens of attributes, the schema includes: information related to the job behavior, e.g. about the number of waiting or running jobs in each queue; information related to the policy regarding this queue, e.g. limits on the number of jobs; and estimates about the queuing delay of each queue, which steers the WMS load balancing algorithm. The complete schema of the IS is available as the GLUE information model; up to now, the IS implements GLUE 1.3 [12]. The information system is conceptually unique, even if its implementation is distributed. Thus, these traces cover the whole EGI infrastructure.

The IS is not natively logged. We thus had to create a logging system of the IS. According to [14], more than 97% of the changes are confined to 14 attributes only. To cope with this massive redundancy, the logging process of the IS is

TABLE II
RTM CATEGORICAL ATTRIBUTES

Name	Description
Jobid	The glite Job identifier
Type	A type computed from the RTM data
FINAL REASON	Job termination status
FINAL EXIT CODE	Job exit code
RB	Name of the Resource Broker
UI	Name of the User Interface
CE	Name of the Computing Element
WN	Name of the Worker Node
VO	Name of the Virtual Organization
DN	User Identity
Requirements	The job requirements
Rank	The ranking formula
RegistrationTimeString	Registration date

TABLE III
RTM TIMESTAMPS

Name	Description: Time at which the job was ...
Userinterface_regjob_Epoch	registered in epoch format
networkserver_accepted_Epoch	accepted by the network server
workloadmanager_match_Epoch	matchmaked to a resource
Jobcontroller_transfer_Epoch	transferred to the resource
logmonitor_accepted_Epoch	seen accepted by the resource by
logmonitor_running_Epoch	seen running by the logmonitor
logmonitor_done_Epoch	seen done by the logmonitor
lrms_running_Epoch	seen running by the lrms
lrms_done_Epoch	seen done by the lrms

realized as follows. The Information System trace uses LDIF (LDAP Data Interchange format), which is the standard ASCII format for representing LDAP directory contents. Each day, a reference complete snapshot is created, and the difference (diff) with this original file is recorded each 15 minutes.

3) Logging and Bookkeeping (L&B): this service logs most of the events in a job's lifecycle, as provided by the various services of gLite. The events are processed to give a higher-level view on the job states (e.g. Submitted, Running and Done when the jobs starts and stops execution, or Transfer from a software component to another one), and records various attributes (e.g. submission file in Job Description language, destination Computing Element name, job exit code, etc.).

Each LB service is associated to a particular WMS; thus, each L&B log covers only those jobs that were managed by this particular WMS. Currently, the GO provides the L&B of only one of the most active WMS; nevertheless, the scope is the full EGI resources.

The native format of the L&B is an SQL database. The trace is provided as a weekly ASCII dump of the tables. It can be easily reloaded as a mySQL database. The most important tables are *events*, *short_fields*, and *long_fields*. Table *events* records the gLite job unique identifier, the number of the event in the job lifecycle, the type of the event, the service generating this event, timestamp and user identifier. The sequential ordering created by event numbering is close to the physical time ordering, but not necessarily identical. The event number is a major index, which allows retrieving information from the two other tables. Table *short_fields* records the internal sub-components of each event. Besides the job identifier and event number, its attributes (columns) are the name (38 possible names) and value of this sub-event. For instance, REASON (name), when associated to a scheduling event, can denote success (e.g. *Job terminated successfully*), or an operational view of the reason of a failure (e.g. *Cannot plan*); SRC_INSTANCE (name) describes, a service, SRC_HOST a physical host etc., the associated value being the name of the service or the host. Table *long_fields* mainly represents information as a blob. For each event, the first row is the JDL of the job, enriched with the default values when the user did not specify a field; the next rows translate and enrich this description in the dialect of the next services the job has to pass through.

4) Accounting: These traces report the information recorded by the batch system of a site, such as scheduling events and memory consumption. Currently, the GO records the logs of the Torque/PBS manager of GRIF (Grid for Research in Ile de France) site. With 6000 cores, 2 PBytes of storage, and a typical usage rate close to 100%, GRIF is the second largest and most active site of EGI in France. The GRIF activity is exclusively gLite, but includes DIRAC-managed jobs.

The traces are provided as daily ASCII files (inside weekly archives, as explained above). A comprehensive presentation of the PBS format, together with scripts for converting these logs into Standard Workload Format, is available on the parallel workload archive project [8].

5) Internal Logs: this category of traces covers various services that take part in the job submission chain. These traces log the internal details of the service activity (table IV). Their main potential usage is diagnosis. The scope is the same as for the L&B. Unfortunately, the formats of these logs are not documented in the gLite suite; this difficulty will be discussed in section III-C1.

6) GridFTP: The GridFTP traces cover the file traffic as far as the GridFTP protocol is used. GridFTP is an extension

TABLE IV
INTERNAL LOGS

Service	Description
Wmproxy	Job submission log: date, user, job identifier.
Jobcontroller	Forwarding of job submission and control requests to Condor-G.
CondorG	Condor-G logging: job submission to the site, executing host and job's status change notification
Logmonitor	Services interactions within the WMS. Relays changes in the state of the job, as obtained from the Condor-G user log, to the rest of the WMS system.
WorkloadManager	Summary of the matchmaking (CE selection), including the number of matching resources, the service time, and the outcome (selected CE).
Jobmap	Gatekeeper information, provides the translation between gLite information and local information.

of the standard File Transfer Protocol (FTP) providing faster transfer and built-in security. It has been standardized by the Global Grid Forum. GridFTP is by far the most heavily used protocol within EGI, irrespective of the middleware. The traces record the successive steps of an FTP transaction (open, close, connect etc.). Amongst the most important information present are the user's identifiers, the name of the machine that issued the request, the direction of the request (get or put), file name, transfer size, outcome (success or failure) and timestamps. These traces exploit the logs of the GridFTP servers at GRIF. The format follows the general syntax of FTP logs.

C. Data models

The second priority of the GO, after data collection, is to provide parsimonious and informative representations of the traces. A first step is converting the traces towards user-friendly formats, in other words taking the burden of the "80% preprocessing" of data mining [1]. While some traces are natively organized along standards (e.g. the IS is GLUE compliant), other ones use a very specific organization of the information (e.g. L&B), proprietary formats (Accounting), and are in the worst case fully undocumented (internal logs of the WMS). To maximize the added value of the costly conversion process, the target organization and format should a) be a (de-facto) standard and b) come with an exploitation framework.

1) An example with CBE: A de-facto standard for the representation of event-oriented traces is the Common Base event (CBE) from IBM. CBE as a format and associated technologies (automatic analysis engine, visualization tool) are the result of IBM's extensive experience with autonomic management. CBE is not suitable for all traces, e.g. the IS, or the jobmap trace in the WMS scope, are not event-oriented. However, CBE adequately covers many of the traces: in the WMS scope, Wmproxy, Jobcontroller, CondorG, Logmonitor and Workloadmanager; outside, the L&B.

The CBE XML schema defines the format of an event, which core is the 3-tuple (reporting component, impacted component, situation). Component may be hardware or software. Twelve generic situations are available, e.g. Start, Stop, and the very important Feature and Dependency ones describing component availability. As a test case, we considered the Jobcontroller logs of GRIF-LAL site from 2008-09-16 to 2009-03-24, with 883,701 events amongst which 118,191 are associated to one or several identifiers. The main challenge is to type each logged event into a situation. The CondorG log format being un-documented, we developed a software suite for elucidating its syntax and to some extent its semantics. The syntax is "class::function:message"; we identified 7 different classes, and 21 functions. Next and more importantly, the messages have been segmented, which required extracting variable strings such as URL or process identifiers, and performing functional inference, in order to type the messages along the CBE thesaurus. For instance, the event "ControllerLoop::run(): Aborting daemon..." will be classified as a CBE "StopSituation" with arguments "successDisposition=SUCCESSFUL" and "situationQualifier=ABORT INITIATED".

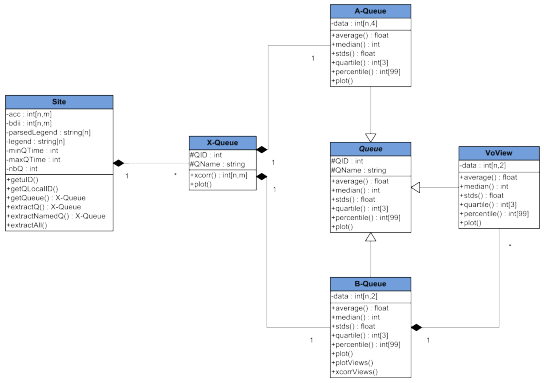

Fig. 3. CE time-series integrated from IS and accounting

Overall, the 883,701 events of the log were finally organized along 7 classes, 21 functions, and 67 messages. Given the size of the traces, it is very likely that a complete data representation has been obtained for this log.

2) Time-series models: More elaborated models are designed ad hoc for serving specific needs. A general requirement is to represent data as time series. In a series model, the focus shifts from the categorical features (event) to the numerical ones (typically timestamps), while categorical features may be present. While a standard has been set for the representation of statistical information and time series for economic data [20], no equivalent concepts have been proposed for the scientific context. We give here an example that highlights the need for integrating sources from two different raw traces.

The evaluation of the performance of the gLite scheduler (see section IV-A) needs to compare the Expected Response Time (ERT) published by the CEs and registered in the Information System, and the Actual Response Time (ART), which is the actual queuing delay experienced by a job at its target CE, and is provided by the Accounting trace. Perl scripts use regular expressions to extract ERT from the patched (reciprocal of diff) IS logs, and ART from the scheduler logs. Here the cost of pre-processing is obvious: the information related to the ERT in one queue represents 0.001% of the total IS data. Next, we use the Object capability of MATLAB to create an efficient data representation with flexible functionalities (fig. 3): the Queue Class represents CE or VOView and provides elementary statistics. Matching Queues are aggregated (Accounting vs IS) in X-Queue, a conceptual Class hosting analysis methods.

The most important method matches the irregular arrival process at the CE and the periodic (900 seconds) sampling of the IS through either interpolation (each job is associated with the ERT of the 900 seconds interval it belongs to), or subsampling (jobs are subsampled at 1mHZ); other useful methods include advanced plotting, cross-correlation, and outliers management. The Site Class groups X-Queues from the same origin, to enable data processing from multiple CEs.

3) Discussion: Whatever more usable representations can be derived, recording and publishing the raw data remains mandatory. Indeed, given the complexity of the recorded data, conversion to a more usable format will have to drop some information. For instance, the replication of the Logmonitor and LRMS timestamps about the same event in the RTM data might seem redundant; it turned up to be the decisive information for failure diagnosis in a clustering-based analysis of this class of traces [26].

IV. Exploiting the GO data

The second role of the Grid Observatory is to contribute to advancing issues that are critical for the future of the EGI grid. The related issues are not technical, but participate in the emergence of Autonomic Computing systems. Amongst them, the GO has recently addressed reinforcement-learning based elastic resource provisioning and decentralized SLA enforcement [16], and dependability through Data Streaming [26]. Two other examples related to the same areas are presented here. The first one is related to scheduling: evaluating the quality of the prediction that controls the gLite fair-share and load balancing strategy. The second example addresses a generic issue in Data Streaming. Data Streaming is the specialized area of Data mining concerned with the real-time exploitation of large data flows. Besides requiring at most linear computational complexity algorithms, the data flow cannot be considered as generated from a stationary distribution; there are changes in the system under study or its environment. The example presented here addresses the detection of abrupt changes (as opposed to trends). The specific goal is to implement a fully automated detection of a specific type of anomalous situation, namely blackholes.

A. Evaluation of a gLite core component

1) Problem statement: gLite maps jobs to CEs by considering the requests in the JDL, and breaks ties (distinguishes between convenient CEs) based on the Estimated Response Time (ERT), an estimation of the queuing delay published by the CEs in the IS. The question is: how good is the ERT predictor? Despite its apparent simplicity, the question is far from trivial. On the technical side, comparing the ERT and ART time series requires to integrate two different traces. As explained above, the IS information was continuously discarded, thus the first GO contribution is to have created the corresponding trace (section III-C2). The resulting experimental dataset corresponds to the Accounting and IS traces from Oct. 2008 to Feb. 2009.

The second issue is the quality of the information. The process described so far is not fully satisfactory: 1) the fixed frequency of IS logs is not adapted to the varying job arrival rate, and 2) the IS ERT might not be the value actually used by the WMS. The first issue requires either interpolating or subsampling, as explained in Section III-C2. The second one is related to the hierarchy of information storage required for making the IS scalable. Each WMS caches a copy of the ERT values from the IS (together with other information), and refreshes it periodically, in order to reduce the pressure on the IS databases. As a consequence, the ERT that was actually used by a job is not logged and had to be estimated. With the information we had, we decided to estimate the ERT of a job by the last published ERT for its target CE.

TABLE V
Statistics for the Atlas and Biomed queues. All times are in seconds.

	Atlas		Biomed	
	ART	ERT	ART	ERT
Mean	1.33E3	2.74E4	3.01E2	2.66E2
Median	11	1	11	1
Std	1.09E4	7.41E4	4.33E3	5.99E3
RMSE	7.94E4		7.21E3	
$q_{90\%}$	1.35E2	1.16E5	25	4
Over. fraction	22%		3%	
Over. median	9.34E4		228	
Under. fraction	77%		96%	
Under. median	9.01E0		9.00E0	

2) Methodology: The main issue for evaluation, however, lies in defining the performance indicator, or objective function. We considered the following four indicators:

- Root Mean Square Error (RMSE). If X is the true value (here the ART) and $\hat{X}$ the predictor (ERT), RMSE $= \sqrt{E[(X-\hat{X})^2]}$. The RMSE is the most elementary and widely used indicator for statistical prediction.
- Similarity in the distributions. This can be evaluated through statistical tests, e.g. Chi-squared, and visually inspected through quantile-quantile plots.
- Cross-correlation. The cross-correlation of the time series $\hat{X}$ and X at lag m is $E[\hat{X}.\text{Shift}(X,m)] = \sum_{n=-\infty}^{+\infty} \hat{X}_n X_{n-m}$.
- The ROC (Receiver Operating Characteristic) curve. This indicator is detailed below. In essence, it estimates the cost/benefit ratio of a predictor, that is the relation between its accuracy and its sensitivity.

The Batch Queue Predictor (BQP) initiative [2] accurately showed that synthetic indicators as RMSE, or the correlation coefficient, may be misleading. RMSE does not provide any information about the localization of the discrepancies along time and along the value range. Distribution similarity gives a better indication, as various segments of the distribution can be inspected, thus localizing in the value range, but does not describe the dynamics. The evaluation criteria should integrate both the statistics about the reliability of the estimation (BQP concept), and the difficulty of the task, which is essential given the highly irregular load profiles: for instance, the (accurate) estimation of a null ERT for an empty CE is not very informative. We will show that the ROC evaluation is compatible with these two goals.

3) Experimental results: We considered only the two most loaded queues (CE-queues), Atlas and Biomed, accounting for some 90% of the overall load. A few suspicious values in the ERT series, which are error codes, have been identified and removed, as well as two obvious outliers in the ART series.

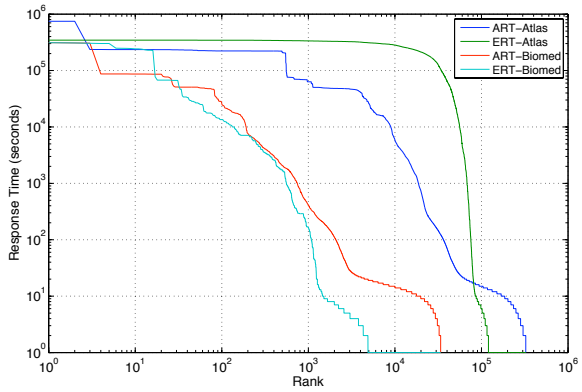

Fig. 4. Distributions of response time

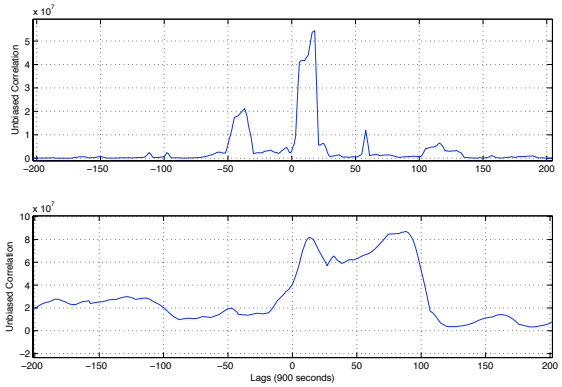

Fig. 6. Cross-correlation. Upper graph: Biomed; Lower graph: Atlas

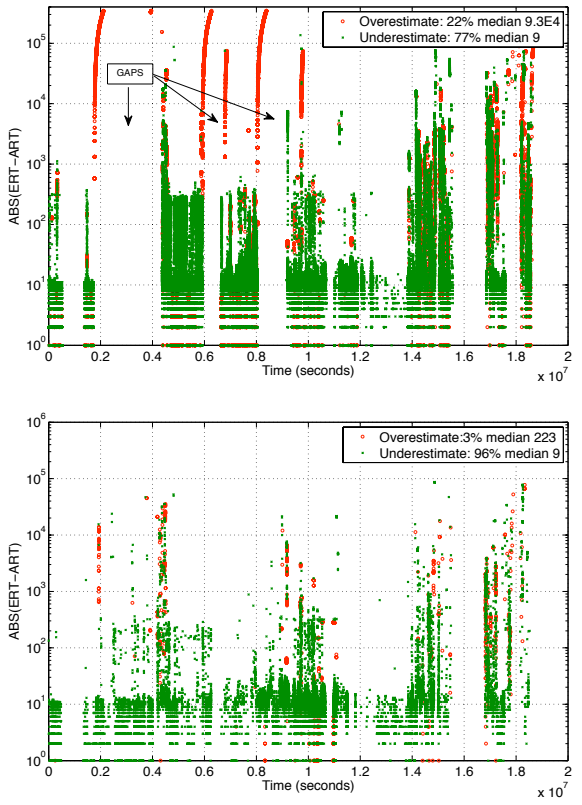

Fig. 5. Absolute values of the error - upper graph: Atlas, lower graph: Biomed. The x axis is the arrival date.

The first four lines of table V give the elementary statistics and the RMSE, using the interpolated ERT. Whether comparing the RMSE with the mean or the median of the ART, the results are extremely poor: the RMSE is at least one order of magnitude larger than the mean.

A closer inspection of the distributions gives the interpretation of the very large RMSE. Fig. 4 is a rank plot of the distribution: the values are plotted against their rank. This plot is similar to a frequency plot, with the x and y axis exchanged. We will not try to fit the distribution, which seems to exhibit some power-law behaviour, as this is relatively irrelevant for our goal. The normal regimes might be similar (or not), the tails are very different. The details of the normal regimes (seconds) do not matter; three orders of magnitude for the 90% percentile for the Atlas queue do matter (fifth line of table V), and explain the massive error in the RMSE. As can be expected, the χ^2 test rejects the hypothesis of similarity: 4 degrees of freedom, p-value = 0 for Atlas, 1 degree of freedom, p-value = 0.2112 for Biomed; the quantile-quantile plots are so unbalanced that we do not show them.

The next step is to explore the dynamics of the phenomenon. Fig. 5 shows the absolute value of the errors, with overestimation (ERT $>$ ART) and underestimation (ERT $<$ ART) plotted with different colors and shapes. Here too, the ERT data are interpolated. The four last lines of table V give the fraction (over the total number of jobs) of over and underestimation, together with the median. While underestimation is by far the most frequent case, its impact should be modest, as the median is negligible. On the other hand, the overestimation is relatively rare, but happens in spikes. The three major ones for Atlas, at times approximately 2×10^6, 6×10^6 and 8×10^6, are followed by a period of inactivity. The comparison with the Biomed case shows that this is not due to a site misbehaviour, as the Biomed queue shows some activity in the same periods. The most likely explanation is that the overestimation of the ERT leads the WMS, or possibly the smart users, to go away from this supposedly overloaded queue.

Fig. 6 displays the cross-correlation as defined above, measured on the subsampled data, which are thus approximately evenly spaced in time. Given the very large sample size, the estimation of the expectation from a finite sample is not an issue, but should be normalized; we used the unbiased estimator $R_{\text{unbiased}}(m) = \frac{1}{N-|m|}R(m)$. The maximum of the correlation appears at lag 89 (approximately 22 hours) for Atlas, and at lag 39 (approximately 10 hours) for biomed. These lags are very large, even if considering the second maximum of Atlas, at lag 64 (16 hours).

As advocated by BQP, the real question is the users' satis-

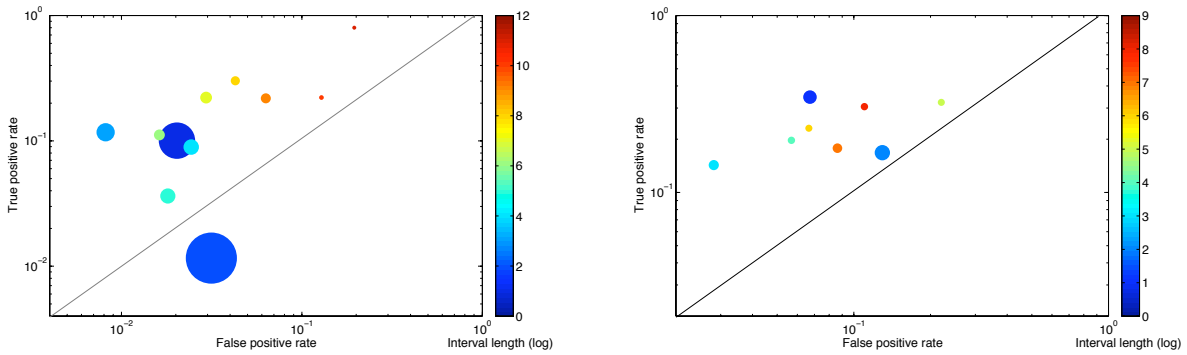

Fig. 7. ROC Analysis - left graph: Atlas, right graph: Biomed

faction in term of precise prediction. The challenge is to have a principled definition of precision. For this, let us first make a detour to binary classification, where the actual and predicted values are binary ("yes/no", or "0/1"). For a binary classifier, the qualitative concept of precision can be technically defined by the accuracy and sensitivity. Let TPR be the rate of true positives, that is the ratio between the number of predicted positives over the number of actual positives, and FPR the rate of false positives, that is the ratio between the number of predicted positives over the number of actual negatives. TPR measures the accuracy, and 1-FPR the sensitivity of the classifier. When the goal is comparing classifiers, the receiver (or relative) operating characteristic (ROC) [13] curve plots TPR as a function of FPR, for each considered classifier. The best case is the (0, 1) point, where all predictions are correct; the worst one is (1,0), where all predictions are erroneous; the diagonal separates the upper region, where the prediction is better than random, from the lower one, where the prediction is worse than random.

The ERT can be considered as a classifier: for a given time interval, the ERT value says whether the waiting time will fall inside this interval (call it a positive), or not (call it a negative). The same association can be done for the ART. A set of intervals will define as many ERT-based classifiers, and each interval will give a point in the ROC space. The question is now the choice of the intervals. Equally spaced intervals have little relevance, both from theoretical analysis, and for the user. One option could be to identify the threshold for a Generalized Pareto distribution (extreme events distribution), then define the intervals as the bins containing the same number of data above this threshold; fit the distribution for the "normal" regime (below the threshold) and define equally sized bins in this region. In order to be not too dependent of a particular data set, we simply defined the intervals bounds as a geometric progression, yielding an exponential growth in the interval size. The upper bound of the series is the maximum of the actual values. Fig. 7 shows the ROC space; in these figures, the surface of the dots is proportional to the actual population of the interval, and the sidebar gives the size of the interval. Overall, the ERT can be considered as a good classifier in most of the cases, as being clearly in the upper region. In the Atlas CE case, there is a large mispredicted interval and FPR increases with the interval size. The main reason is the spike effect described above, which populates the highest interval with erroneously large ERTs. In the Biomed CE case, there is no obvious relation between FPR and interval size, due to a much less pronounced spike effect.

$$\overline{x_t} = \frac{1}{t}\sum_{l=1}^{t} x_t$$
$$m_T = \sum_{t=1}^{T}(x_t - \overline{x_t} + \delta)$$
$$M_T = \max\{m_t; t = 1 \dots T\}$$
$$\mathrm{PH}_T = M_T - m_T$$

Fig. 8. Computation of the PH statistics for process x_t

B. Mining the Jobs Stream

1) Problem statement: The matchmaking process enacted by the WMS may occasionally result in a CE or a site turning to a *blackhole*. A blackhole site or CE executes jobs at very high rate; the execution is actually faulty, however the middleware may not report the fault, depending on its cause. As a consequence, the CE or the site reports high availability and drains the grid jobs only to fail them. The challenge is early and automated detection of blackholes.

2) Method: Here too, the question seems simpler than it actually is. Indeed, properly functioning CEs may experience a surge in job submission, mainly because of compound jobs or data challenges involving many short jobs, coupled with users' selection of this CE. As the goal is to detect abrupt changes, the Page-Hinkley (PH) statistics (fig. 8) provides an efficient method, which provably minimizes the time to detection for a prescribed false alarm rate. The PH test detects jumps in the mean, and triggers an alarm when the PH statistics exceeds a threshold λ. The efficiency of the test, compared to naive thresholding, comes from the integration of the history of the process in the statistics (sequential testing [23]). The parameter controls the false alarm rate, while the δ parameter controls the memory (or obliviousness) of the statistics.

3) Experimental results: We have applied this method to quantities (number of arrived and served jobs per unit of time)

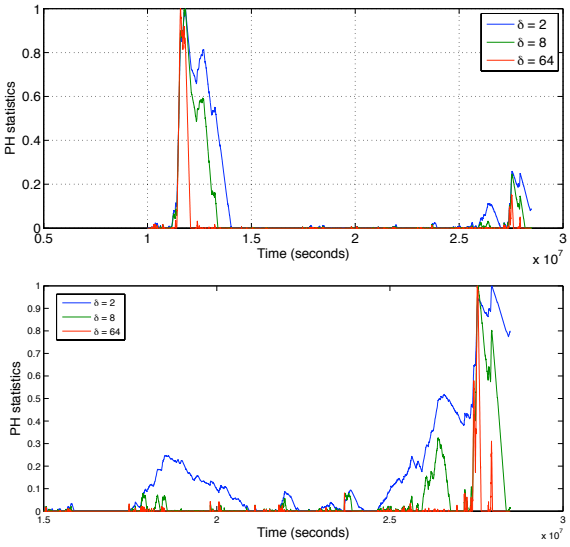

Fig. 9. PH statistics for the number of arriving jobs. The statistics is normalized for easier visualisation, by setting all maxima to 1. Upper graph: PH over the whole period. Lower graph: hardware-induced blackhole.

that are easily computed from the accounting trace, and can be computed in real-time from the corresponding log. The main result is that we are able to efficiently detect blackholes of very different origins without human tuning.

Fig. 9, upper graph, shows the PH statistics for the arrivals (the service one being very similar is not shown). The δ parameter is usually set up empirically, thus it is varied here along $\delta = 2^i$, $i = 0 \dots 8$, in order to pick-up reasonable values. Note that this is to be done once, for the general parameter setting. We have added the $\delta = 0$ test, only to show that it cannot be sensitive enough. The threshold must be calibrated on a reference period of normal (but not overly low) activity. This period should be initially defined by the system administrator. Here, the first 10^4 seconds are known to correspond to a normal activity. The threshold λ is thus set at the maximum of the monitored quantity in the interval $[0, 10^4]$. The *first alarm* (earliest time after the reference period where the threshold is violated) happens at time 1.1433×10^7, with a standard deviation over $i = 1 \dots 8$ of 1.68×10^3 seconds. The same procedure applied to the number of served jobs gives the same first alarm, but with a slightly higher standard deviation of 1.98×10^3 seconds. Based on this case and the next example, covering two typical and different blackhole situations, $\delta = 64$ is the best choice, although all values are in fact acceptable.

Naive thresholding would simply threshold the instantaneous rates, possibly smoothed. For the sake of completeness, we show in details that this policy is fully unable to give equivalent results. We first experiment the classical "3σ" method, where outliers are the data above three standard deviations of the past process. In the following counter-experiments, the distribution parameters are extracted from the past at each instant of the dataset Without smoothing, the results are fully inconsistent; after gaussian smoothing, the results (corrected from the filter) are inaccurate: the mean over reasonable smoothings is 1.1206×10^7 with a standard deviation of 1.72×10^4 seconds. The failure of the "3σ" outliers detection was to be expected, as the distributions of the arrival and service times are anything but normal. A more principled method is to fit a Generalized Pareto distribution to the tail of the actual dataset. With the same smoothings, the results are even more dispersed, with a standard deviation at 5.7600×10^4 seconds. This illustrates a general requirement for many (mis)behaviour run-time tests, which is to identify if they target abrupt changes. In this case, short-term history matters, not the general distribution parameters.

This experiment also illustrate the need for expert evaluation in the analysis process. It turns out that the system administrators did not identify the dominating peak as a blackhole, as the blackhole-like behavior does not come from the site hardware or middleware, but from a software error of an HEP experiment. In a real-world situation, the administrator would be alerted, run tests to check the proper functioning of the site, and probably warn the erroneous user group; it would then restart the statistics gathering process. Simulating this simply amounts to restart the PH test after the software-induced blackhole, with λ unmodified. Fig. 9, lower graph, shows the PH normalized over the next period. The peak was also visible on the left graph at a much smaller scale. The first alarm is at 2.7306×10^7, with a standard deviation of 93 seconds, and corresponds to a hardware-induced blackhole.

V. Related Work

We first refer the reader to an extensive analysis of the major computer science and engineering archives presented in the paper introducing the Grid Workload Archive (GWA) [5]. The Grid Observatory approach is obviously related to the initiatives towards collecting grid traces such as GWA or more recently the Failure Trace Archive [17] in publicly providing traces of grid activity. The major differences are in scope and comprehensiveness.

Scope While most available traces at GWA come from computer science research grids, the GO source is the EGEE/EGI production grid. With application domains including high-energy physics, bioinformatics, computational chemistry and many more, EGI provides a good approximation of the requirements and behavior of the overall community of e-science users. Two facts follow: first, the bulk of the activity reported by the GO is high-throughput computing, instead of parallel applications as reported in GWA; second, as remarked in [10], the workload is much higher.

Comprehensiveness So far, the existing repositories are explicitly limited to 1) sparse sampling and 2) specific scope (e.g. scheduling). On the contrary, the GO goal is to provide 1) continuous data series and 2) comprehensive coverage of the usage and middleware activity, as discussed in section III-B. The primary motivation for the first goal is that we simply do not know enough to assess the significance of any particular segment of the grid activity. Section III-C2 has shown one small example of the need for integrating multiple sources

of information. More generally, comprehensiveness might be critical for achieving the general goal of disentangling the intrinsics of users behavior from the middleware processing (analogous to data locality vs cache hit or miss) [15].

Common knowledge in the area of data curation says "Services Make the Repository". Indeed, Internet-related archives such as the Internet Traffic Archive or CAIDA offer extensive community-based toolkits for trace selection, feature extraction, basic analysis, and visualization. With the goal of integrating its heterogeneous data source, the next step for the GO would be building a knowledge representation facilitating the usage of the GO data. State-of-the art knowledge management is based on ontology. The ontology should integrate, when appropriate, existing grid ontologies (structural concepts), but extend them in the directions of new concepts describing grid inputs and dynamics. Examples of Grid ontologies are the Grid Organizational Memory [6], the CoreGrid Ontology [24] and the Grid Resource Ontology (GRO). They intend to represent both general knowledge about grids in a top-level ontology and knowledge dedicated to specific grids in sub-ontologies. However, though they introduce a complementary structuration level (modular ontologies), they remain lightweight ontologies with very limited inference capacities.

VI. Conclusion and Future Work

The Grid Observatory integrates a production service for the CSE community, and a research project of building grid knowledge. The grid infrastructure, supported by EGEE and other projects, has evolved into EGI, which is managed by a independent legal entity, EGI.eu and controlled by its member National Grid Initiatives and Associated Participants, guaranteeing the long-term availability of the grid infrastructure to its users. This commitment motivates the GO ambitious and long-term data curation project. The GO user community is steadily increasing: the availability of e-science representative data seems to be a sufficient motivation to overcome the *relative* difficulties associated to providing only raw traces.

The general framework for the GO future work is to turn it into a social intelligence system to pool scientific and engineering expertise, in order to build gradually more integrated models of the e-infrastructures, and to define and validate advanced, autonomic-oriented policies addressing the operational challenges of the production European e-infrastructures.

Acknowledgments

This work has been partially supported by the European Infrastructure Projects EGI-InsPIRE INFSO-RI-261323.

References

[1] P. Adriaans and D. Zantingue. *Data Mining*. Addison-Wesley, 1996.

[2] J. Brevik, D. Nurmi, and R. Wolski. Predicting bounds on queuing delay for batch-scheduled parallel machines. In *11th ACM SIGPLAN symp. on Principles and Practice of Parallel Programming, PPoPP '06*, pages 110–118, 2006.

[3] A. Cady and C. Germain-Renaud. The GO technical documentation. http://query.grid-observatory.org/GOTechnicalDocV1.2.pdf.

[4] T. Elteto, C. Germain Renaud, P. Bondon, and M. Sebag. Discovering Piecewise Linear Models of Grid Workload. In *10th IEEE Int. Symp. on Cluster, Cloud and Grid Computing*, May 2010.

[5] A. Iosup et al. The Grid Workloads Archive. *Future Gener. Comput. Syst.*, 24(7):672–686, 2008.

[6] B. Kryza et al. Grid organizational memory-provision of a high-level grid abstraction layer supported by ontology alignment. *Future Gener. Comput. Syst.*, 23:348–358, 2007.

[7] D. Colling et al. Real time monitor of grid job executions. *Jal. of Physics: Conf. Ser.*, 219(6), 2010. RTM project: http://gridportal.hep.ph.ic.ac.uk/rtm/.

[8] D. G. Feitelson et al. The parallel workload archive. http://www.cs.huji.ac.il/labs/parallel/workload/.

[9] E. Laure et al. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006.

[10] I. Rodero et al. Energy-Efficient Application-Aware Online Provisioning for Virtualized Clouds and Data Centers. In *Int. Green Computing Conf. (IGCC 2010)*, 2010.

[11] M. Ruda et al. A uniform job monitoring service in multiple job universes. In *Workshop on Grid monitoring, GMW '07*, pages 17–22. ACM Press, 2007.

[12] S. Andreozzi et al. Glue Schema Specification, V1.3. Technical report, Open Grid Forum, 2008.

[13] T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27, June 2006.

[14] L. Field and M.W. Schultz. An investigation into the mutability of information in production grid information systems. *Jal. of Physics: Conf. Ser.*, 219(6), 2010.

[15] R. Fonseca, V. Almeida, and M. Crovella. Locality in a web of streams. *Commun. ACM*, 48:82–88, 2005.

[16] C. Germain-Renaud, J. Perez, B. Kégl, and C. Loomis. Multi-objective Reinforcement Learning for Responsive Grids. *The Journal of Grid Computing*, 8(3):473–492, 2010.

[17] D. Kondo, B. Javadi, A. Iosup, and D. Epema. The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems. In *10th IEEE/ACM int. Conf on Cluster, Cloud and Grid Computing, CCGrid 2010*, pages 398–407, 2010.

[18] A. Luckow, L. Lacinski, and S. Jha. Saga bigjob: An extensible and interoperable pilot-job abstraction for distributed applications and systems. In *10th IEEE/ACM int. Conf on Cluster, Cloud and Grid Computing, CCGrid 2010*, pages 135–144, 2010.

[19] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *7th IEEE Int. Symp. on High Performance Distributed Computing*, pages 28–31, 1998.

[20] The Statistical and Metadata eXchange Initiative. SDMX-EDI : Syntax and Documentation. Technical report, 2004. http://www.sdmx.org/.

[21] A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees. DIRAC: A Scalable Lightweight Architecture for High Throughput Computing. In *5th IEEE/ACM Int. Workshop on Grid Computing (GRID'04)*, 2004.

[22] V.Korkhov, J. T. Moscicki, and V. V. Krzhizhanovskaya. Dynamic workload balancing of parallel applications with user-level scheduling on the grid. *Future Generation Comp. Syst.*, 25(1):28–34, 2009.

[23] A. Wald. *Sequential Analysis*. Wiley Series in Stat. Wiley, 1966.

[24] W. Xing, M. D. Dikaiakos, and R. Sakellariou. A Core Grid Ontology for the Semantic Grid. In *6th IEEE/ACM int. Conf on Cluster, Cloud and Grid Computing, CCGrid 2006*, pages 178–184, 2006.

[25] E. Yakel. Digital curation. *OCLC Systems & Services.*, 23(4):335–340, 2007.

[26] X. Zhang et al. Toward Autonomic Grids: Analyzing the Job Flow with Affinity Streaming. In *15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 987–996, 2009.